Combining SLA Prediction and Cross Layer Adaptation for Preventing SLA Violations

Eric Schmieders[1], Andras Micsik[2], Marc Oriol[3], Khaled Mahbub[4] and Raman Kazhamiakin[5]

[1] Paluno, University of Duisburg-Essen, Essen, Germany
eric.schmieders@paluno.uni-due.de
[2] MTA SZTAKI DSD, Budapest, Hungary
micsik@sztaki.hu
[3] Universitat Politècnica de Catalunya, Barcelona, Spain
moriol@lsi.upc.edu
[4] SOI (CITY University), London, United Kingdom
k.mahbub@soi.city.ac.uk
[5] FBK-Irst, Trento, Italy
raman@fbk.eu

Abstract. Service-based Applications (SBA) are deployed in highly dynamic and distributed settings, where various parts of the constituent components - services and their infrastructure - are controlled by different third parties. In such a loosely coupled environment, adaptation capabilities are needed to manage deviations and unforeseen situations which might lead to negative consequences (e.g. contractual penalties). Current approaches either focus on cross-layer-adaptation or the prevention of SLA violations. In contrast to this, the approach presented in this paper combines both. The paper presents an architecture as a generic framework for the management of arising problems during service execution. Multiple adaptation mechanisms are available to react on adaptation needs, acting on different layers of the SBA (including e.g. the composition layer and the infrastructure layer). The final goal of the cross-layer adaptation capability is to avoid the violation of agreed Service Level (in SLAs) and thus ensure the benefits of SBAs for both customers and providers.

1 Introduction

Service-based Applications (SBAs) provide their functionality in a rather dynamic way using a number of possibly independent services in a loosely coupled way based on the paradigm of Service-oriented Architecture (SOA). The main difference between a traditional modularly built application and an SBA is that the latter cannot control fully its components, but relies on external services provided by different parties. An SBA typically operates on different levels in order to implement its functionality. On the highest level, the performance and quality of offered services as a whole is the main issue, which breaks down into the issue of composing services and executing service compositions. On a lower level, each

service of a composition has to be discovered, selected and executed, which are the basic and classical service provisioning functions. Some services may run at a third party (the case of outsourcing), while some services are executed in a local infrastructure or in a Cloud environment. In latter cases, the platform and proper environment for the execution have to be prepared and maintained for service executions.

As studies reveal (e.g. [12]), distributed applications are extremely fragile, due to the execution of uncontrollable external services. Unexpected changes of third party services or unpredicted network latencies, for example, can cause failures avoiding the timely execution of the SBA. In consequence, failures can cause violations of the agreed Service Level Agreements (SLAs), what might lead to further negative consequences, e.g. contractual penalties. In order to prevent SLA violations, adaptation capabilities are needed to react to failures and unforeseen situations during the execution before the SLAs are violated.

Current approaches either focus on performing cross-layer-adaptation in a strictly reactive way, hence not avoiding SLA violations and the implicit contractual penalties. Or they aim to prevent SLA violations, by focusing on the Service Composition and Coordination Layer (also known as SCC, cf. definitions in [9]). Those approaches don't exploit every opportunity to prevent an SLA violation and thus face situations where a prevention is not possible, e.g. when internal services are to be invoked which cannot be simply replaced. To address this gap, the paper describes a framework which prevents SLA violations of SBAs, by applying cross-layer adaptation via the cooperation of monitoring and adaptation facilities.

Real-life use-cases demonstrating the need for our approach can be found for example among emergency services where certain reactions have to be made in given time limits, or among financial services where transactions going late may cause money loss (e.g. stock exchange).

The rest of the paper contains an overview of related work on cross-layer adaptation (Section 2), the detailed description of our approach (Section 3), and the conclusion of our work (Section 4).

2 Related Work

In [3] Gjørven et al. propose a framework to support cross-layer self-adaption in SBAs. They present a technologically independent middleware named QUA to support coordinated cross-layer adaptations by integrating interface and application layer adaptation mechanisms. Similarly, Popescu et al. also present a cross-layer adaptation framework in [10] with internal and external services which uses adaptation templates to define the behaviour of adaptation processes. Focusing on particular layers, Vidackovic et al. [11] present a cross-layer monitoring and adaptation framework, where they aim at the relationships of the Business Model with respect to the lower layers, and propose a cross-layer adaptation strategy based on a top-down approach. Although these approaches

handle cross-layering adaptations, they do not avoid the violation of the SLA during the execution of the SBA.

Regarding approaches limited to the service composition layer, V. Cardellini and S. Iannucci [1] present a framework named MOSES with two concrete adaptation strategies, namely service selection and coordination pattern selection. When a service fails, the adaptation is performed only for future invocations of the SBA. For that reason, it does not avoid the violation of SLAs during the SBA's execution. This problem is addressed by Leitner et al. in [4]. Leitner et al. present a framework, named PREvent, in order to predict end-to-end performance violations of SBAs during their execution. In contrast to our work the presented framework needs hundreds of SBA executions to provide precise violation predictions. This is due to the exploitation of machine learning techniques, which in general requires a huge amount of training data for adequate results.

In summary, the above mentioned solutions fall into two disjunct categories: one category facilitates cross layer adaptation, while the other avoids SLA violations of executed SBA. Our contribution combines both aspects: instrumenting cross layer adaptation in order to avoid SLA violations.

3 Approach

The key idea behind the integrated solution represented in this approach is based on the use of assumptions on the SBA's context. The context is not under control of the SBA provider, as it includes e.g. third party services. The important aspect of the assumptions in our approach is, that the assumptions are used to relate the continuously monitored data to the SBA's requirements. In particular, we exploit monitors to check whether the assumptions are still satisfied. In case an assumption is violated, we check immediately whether the end-to-end requirement is violated as well. If the check reveals a violation of at least one requirement, the service composition must be adapted in order to compensate the delay occurred in the preceding execution of the SBA instance. To achieve the adaptation an appropriate strategy is generated by a multi-agent community. Finally, the adaptation strategy must be executed.

The architecture of the solution is represented in Fig. 1. The workflow is executed by a Process Engine that communicates with the services through an Enterprise Service Bus (ESB). The ESB routes the service invocations and serves as an aggregation point for different types of events from different sources (e.g. from service calls). Furthermore, the ESB enables dynamic rerouting of requests to different services. The information about available services and their current properties is stored in the service repository. In the following the components and the relation to the different layers are introduced.

Service Monitoring The key monitoring component is SALMon (cf. [8]). SALMon is used to monitor assumptions regarding the different properties of individual services, specially their non-functional properties (e.g. availability, response time). SALMon is able to (1) monitor the QoS of services in a SBA (Service Composition and Coordination Layer) and (2) check if the retrieved

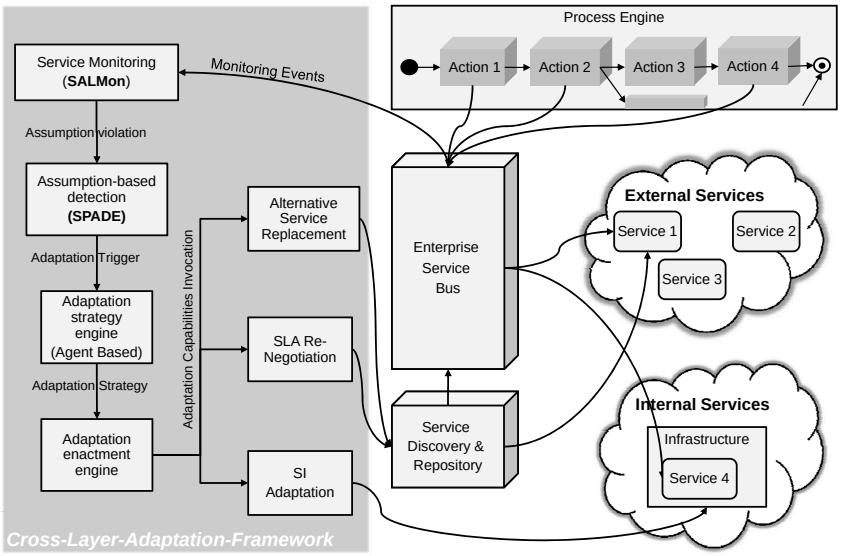

Fig. 1. Architecture

QoS matches with expected values. Furthermore, SALMon is able to monitor process KPIs like the process' end-to-end response time, hence also addressing the Business Process Management (BPM) layer. In order to be easily integrable with other technologies, SALMon has been implemented as an SBA by itself.

Adaptation-based Detection If a violation of the stated conditions occurs, SALMon notifies the Specification and Assumption Based Detection (SPADE) of adaptation needs, whose basics are introduced in [6]. The notification contains the violated assumptions and the violating value.

The SPADE component is used to evaluate the impact of the violated assumption on the corresponding application requirement. SPADE performs a runtime verification of the process model against the requirements. SPADE checks, whether the workflow specification S, the monitored data M and the assumptions in A' satisfy the given requirements R, that is: $S, M, A' \models R$. The set A' comprises assumptions related to services which are not invoked at the point in time when the check is performed.

If R is satisfied, then the workflow execution is continued. If R is not satisfied, the SBA or the service infrastructure (SI) must be adapted in order to compensate the delay. In the latter case the adaptation strategy engine is invoked.

Adaptation Strategy Engine The adaptation strategy engine is manifested as a multi-agent platform where agents implement intelligent behaviors and negotiations with each other in order to collect available information and make decisions on adaptation strategies. Each service composition instance is represented by a Process Agent responsible for the proper and timely execution of this instance. Internal and external services are represented by Service

Agents, comprising information of the service properties (like response time) and supported adaptation capabilities.

A Process Agent is instantiated together with the instantiation of each process. SALMon triggers the responsible Process Agent whenever a deviation from the agreed SLA is predicted. In this case, the Process Agent issues requests for an adaptation solution to Service Agents, which, in turn, respond with their offered adaptation solution. The Process Agent chooses one from the offered adaptation strategies. By running the SPADE check the Process Agent assures the adherence to the SBA requirements for the chosen adaptation strategy. This strategy can comprise several invocations of one or more Adaptation Capabilities affecting the SCC and SI layer. Once the Process Agent has chosen an adaptation strategy the Process Agent forwards this strategy to the Adaptation Enactment Engine for execution. This engine executes the strategy invoking the Adaptation Capabilities according to the strategy.

Adaptation Enactment Engine and Adaptation Capabilities The adaptation strategy is executed by the Adaptation Enactment Engine. The adaptation capabilities invoked during the execution are described in the following.

Service replacement When performing the service replacement, the service repository is involved (cf. [5]). In particular, the request for a service replacement is transformed into the query of the runtime service discovery tool. The identification of alternative services is based on various characteristics of the published services such as structural, behavioral and quality characteristics that services should have in order to be acceptable replacements for a constituent service. The actual service replacement is performed via the ESB, which reroutes the service invocations to the replacing service. The service replacement is located at the SCC layer.

SLA re-negotiation The SLA negotiation broker performs the SLA negotiation for each candidate service identified by the service discovery tool, located at the SI-layer (cf. [5]). The desired quality level is negotiated with the selected candidate service. The QoS characteristics of each candidate service are negotiated in order to achieve the best possible SLA for the service that is within the boundary constraints (e.g. costs and response time) of the service provider and the consumer.

SI Adaptation In case of internal services we propose to influence the infrastructure of the internal services. For example, in order to catch up lost time caused by preceding service failures, the execution speed of an internal service could be increased. This is possible, as the process owner has access to the infrastructure of internal services, and can make executions run faster. For this purpose several approaches are available (eg. [7]). The SI adaptation is located at the SI-layer.

4 Conclusions and Future work

In the paper, a solution was presented to avoid SLA violations in the complex settings of Service-based applications, thus addressing a gap in the current lit-

erature which either prevents SLA violations or applies cross-layer-adaptation. The novelty of the approach is the exploitation of all SBA layers (BPM, SCC and SI) for the prevention of SLA violations. The identification of adaptation needs is based on SLA prediction, which uses assumptions on the characteristics of the running execution context. Multiple adaptation mechanisms are available to react on the adaptation need, acting on different layers of the SBA. The adaptation strategy chooses the right adaptation mechanism, coordinated by a multi-agent community. In the future, we plan to further generalize the mechanism for handling the adaptation management cycle of detection, selection and enactment, and to incorporate more components, event types and adaptation capabilities.

Acknowledgments: We cordially thank Andreas Metzger for helpful comments on the paper draft. The research leading to these results has received funding from the European Community's 7th Framework Programme FP7/2007-2013 under grant agreement 215483 (S-Cube).

References

1. Cardellini, V., Iannucci, S.: Designing a broker for qos-driven runtime adaptation of soa applications. In: ICWS. pp. 504–511 (2010)
2. Gehlert, A., Bucchiarone, A., Kazhamiakin, R., Metzger, A., Pistore, M., Pohl, K.: Exploiting assumption-based verification for the adaptation of service-based applications. In: SAC. pp. 2430–2437 (2010)
3. Gjørven, E., Rouvoy, R., Eliassen, F.: Cross-layer self-adaptation of service-oriented architectures. In: MW4SOC. pp. 37–42 (2008)
4. Leitner, P., Michlmayr, A., Rosenberg, F., Dustdar, S.: Monitoring, prediction and prevention of sla violations in composite services. In: ICWS. pp. 369–376 (2010)
5. Mahbub, K., Spanoudakis, G.: Proactive sla negotiation for service based systems. In: Proceedings of the 2010 6th World Congress on Services. pp. 519–526. SERVICES '10, IEEE Computer Society, Washington, DC, USA (2010)
6. Metzger, A., Schmieders, E., Cappiello, C., Nitto, E.D., Kazhamiakin, R., Pernici, B., Pistore, M.: Towards proactive adaptation: A journey along the s-cube service life-cycle. In: Maintenance and Evolution of Service-Oriented Systems (2010)
7. Nallur, V., Bahsoon, R.: Self-adapting applications based on qa requirements in the cloud using market-based heuristics. In: ServiceWave. pp. 51–62 (2010)
8. Oriol, M., Franch, X., Marco, J., Ameller, D.: Monitoring adaptable soa-systems using salmon. In: Workshop on Service Monitoring, Adaptation and Beyond (Mona+). pp. 19–28 (2008)
9. Pistore, M., Kazhamiakin, R., Bucchiarone, A.: Integration framework baseline. Tech. rep. (2009)
10. Popescu, R., Staikopoulos, A., Liu, P., Brogi, A., Clarke, S.: Taxonomy-driven adaptation of multi-layer applications using templates. In: SASO. pp. 213–222 (2010)
11. Vidackovic, K., Weiner, N., Kett, H., Renner, T.: Towards business-oriented monitoring and adaptation of distributed service-based applications from a process owner's viewpoint. In: ICSOC/ServiceWave Workshops. pp. 385–394 (2009)
12. Zheng, Z., Zhang, Y., Lyu, M.R.: Distributed qos evaluation for real-world web services. In: Proceedings of the 2010 IEEE International Conference on Web Services. pp. 83–90. ICWS '10, IEEE Computer Society, Washington, DC, USA (2010)